Amendments to the Specification:

Please replace the paragraph [0021] with the following rewritten paragraph [0021]:

[0021] In the drawings:

FIG. 1 is a block diagram that illustrates compiling a program written in a high-level language.

FIG. 2 is a flow diagram that illustrates stack usage for adding two 16-bit operands widened to 32-bits.

FIG. 3 is a flow diagram that illustrates stack usage for adding two 32-bit operands.

FIG. 4 is a block diagram of a client computer system suitable for implementing aspects of the present invention.

FIG. 5 is a block diagram that illustrates converting arithmetic expressions for execution on a resource-constrained device according to one embodiment of the present invention.

FIG. 6 is a block diagram that illustrates converting Java™ class files in accordance with one embodiment of the present invention.

FIG. 7A is a code sample that illustrates the addition of two values of type "short" on a desktop computer.

FIG. 7B is a code sample that illustrates the addition of two values of type "short" on a resource-constrained computer.

GUNNISON, McKAY & HODGSON, L.L.P.
Garden West Office Plaza
1900 Garden Road, Suite 220
Monterey, CA 93940
(831) 655-0880
Fax (831) 655-0888

FIG. 8A is a code sample that illustrates the addition of two values of type "short" and immediately casting the result on a desktop computer.

FIG. 8B is a code sample that illustrates immediately casting the result of an operation that potentially carries overflow on a resource-constrained computer.

FIG. 9A is a code sample that illustrates the addition of three values of type "short" and immediately casting the result on a desktop computer.

FIG. 9B is a code sample that illustrates performing an operation that is not affected by overflow on operands created by an operation that potentially carries overflow on a resource-constrained computer.

FIG. 10A is a code sample that illustrates the addition of two values of type "short" and dividing the result by a value of type "short" on a desktop computer.

FIG. 10B is a code sample that illustrates performing an operation that is affected by overflow on operands created by an operation that the potential for overflow on a resource-constrained computer.

FIG. 11 is a flow diagram that illustrates a method for n-base typed arithmetic expression optimization in accordance with one embodiment of the present invention.

FIG. 12 is a block diagram that illustrates instruction data maintained during expression optimization in accordance with one embodiment of the present invention.

GUNNISON, McKAY & HODGSON, L.L.P.
Garden West Office Plaza
1900 Garden Road, Suite 220
Monterey, CA 93940
(831) 655-0880
Fax (831) 655-0888

FIG. 13 is a block diagram that illustrates procedure- or method-calling relationships in accordance with one embodiment of the present invention.

FIG. 14 is a detailed flow diagram that illustrates a method for n-base typed arithmetic expression optimization in accordance with one embodiment of the present invention.

FIG. 15 is a flow diagram that illustrates a method for recording input instructions in accordance with one embodiment of the present invention.

FIG. 16 is a flow diagram that illustrates a method for converting an instruction in accordance with one embodiment of the present invention.

FIG. 17 is a flow diagram that illustrates a method for validating input stacks in accordance with one embodiment of the present invention.

FIG. 18 is a flow diagram that illustrates a method for comparing input stacks in accordance with one embodiment of the present invention.

FIG. 19 is a flow diagram that illustrates a method for optimizing an instruction type in accordance with one embodiment of the present invention.

FIG. 20 is a flow diagram that illustrates a method for matching operand types with an instruction type in accordance with one embodiment of the present invention.

GUNNISON, McKAY & HODGSON, L.L.P.
Garden West Office Plaza
1900 Garden Road, Suite 220
Monterey, CA 93940
(831) 655-0880
Fax (831) 655-0888

FIG. 21 is a flow diagram that illustrates a method for changing an operand type in accordance with one embodiment of the present invention.

FIG. 22 is a flow diagram that illustrates a method for changing an instruction type in accordance with one embodiment of the present invention.

FIG. 23 is a flow diagram that illustrates a method for recording results in accordance with one embodiment of the present invention.

FIG. 24A is a flow diagram that illustrates a method for determining potential overflow in accordance with one embodiment of the present invention.

FIG. 24B is a flow diagram that illustrates a method for determining potential overflow in accordance with one embodiment of the present invention.

FIG. 25 is a flow diagram that illustrates a method for generating an output stack in accordance with one embodiment of the present invention.

FIG. 26 is a block diagram that illustrates conversion of an ~~arithmetic~~arithmetic expression that can be optimized to smaller type instructions in accordance with one embodiment of the present invention.

FIG. 27 is a high-level block diagram that illustrates conversion of an ~~arithmetic~~arithmetic expression that cannot be optimized to smaller type instructions in accordance with one embodiment of the present invention.

GUNNISON, McKAY & HODGSON, L.L.P.
Garden West Office Plaza
1900 Garden Road, Suite 220
Monterey, CA 93940
(831) 655-0880
Fax (831) 655-0888

FIG. 28 is a detailed block diagram that illustrates conversion of an ~~arithmetic~~arithmetic expression that cannot be optimized to smaller type instructions in accordance with one embodiment of the present invention.

FIG. 29 is a block diagram that illustrates instruction conversion where input stacks do not match in accordance with one embodiment of the present invention.

FIG. 30 is a detailed block diagram that illustrates instruction conversion where input stacks do not match in accordance with one embodiment of the present invention.

FIG. 31 is a block diagram that illustrates conversion of an instruction that has multiple consumers in accordance with one embodiment of the present invention.

FIG. 32 is a detailed block diagram that illustrates conversion of an instruction that has multiple consumers in accordance with one embodiment of the present invention.

Please replace the paragraph [0062] with the following rewritten paragraph [0062]:

[0062] Turning now to FIG. 16, a flow diagram that illustrates a method for converting an instruction in accordance with one embodiment of the present invention is presented. Figure 16 provides more detail for reference numeral 1420 of FIG. 14. At 1600, the input stacks for the instruction to be converted are received. The number of input stacks associated with an instruction corresponds to the number of input instructions associated with the

GUNNISON, McKAY & HODGSON, L.L.P.
Garden West Office Plaza
1900 Garden Road, Suite 220
Monterey, CA 93940
(831) 655-0880
Fax (831) 655-0888

instruction. By way of example, if a particular instruction has three input instructions, the instruction has three input stacks, each of which is associated with one of the three input instructions. At 1605, a determination is made regarding whether any of the input stacks are empty or nonexistent. If any of the input stacks are null, then the corresponding input instruction has not yet been converted and an indication that another pass is required is set at 1610. If at least one input stack is non-empty, the input stacks are validated at 1615 to ensure that corresponding entries in input stacks have the same types and to ensure that the input stacks have the correct number of operands and type of operands. At 1625, the instruction type is optimized based on instruction inputs to determine the smallest usable instruction type. At 1630, the operand types are matched with the instruction type to ensure that the types of operands in the input stack are compatible with the instruction requirements. At 1635, the results are recorded by performing the operations of the instructions on the stack.

Please replace the paragraph [0080] with the following rewritten paragraph [0080]:

[0080] Turning now to FIG. 26, a block diagram that illustrates conversion of an ~~arithmetic~~arithmetic expression that can be optimized to smaller type instructions in accordance with one embodiment of the present invention is presented. In FIG. 26, the conversion process is applied to the Java™ expression

c = (short) ((short) (a + b) / c)

GUNNISON, McKAY & HODGSON, L.L.P.
Garden West Office Plaza
1900 Garden Road, Suite 220
Monterey, CA 93940
(831) 655-0880
Fax (831) 655-0888

where the values "a", "b" and "c" are of type "short". The Java™ bytecode sequence for this expression is shown at reference numeral 2608.

Please replace the paragraph [0100] with the following rewritten paragraph [0100]:

[0100] Turning now to FIG. 28, a detailed block diagram that illustrates instruction conversion of an ~~arithmetic~~arithmetic expression that cannot be optimized to smaller type instructions in accordance with one embodiment of the present invention is presented. Figure 28 provides more detail for FIG. 27. Sequence 1 (2812) shows the original Java™ bytecode. In sequence 2 (2814), the initial results of conversion render "short"-type instructions. When the divide operation is encountered, an "int"-type divide instruction ("idiv") is generated because one of the inputs ("sadd" result) has potential overflow from a "short"-type value and the divide operation is sensitive to overflow. The "int"-type divide ("idiv") forces each input operand to become type "int". In sequence 2 (2814), this conversion is triggered for the "short"-type addition instruction "sadd", rendering an "int"-type addition instruction ("iadd").

GUNNISON, McKAY & HODGSON, L.L.P.
Garden West Office Plaza
1900 Garden Road, Suite 220
Monterey, CA 93940
(831) 655-0880
Fax (831) 655-0888